WEST Search History

```
DATE:  Friday, September 05, 2003
```

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
L23	L22 NOT l21	12	L23
L22	l18 and ((quality adj4 service) or QoS or QOS or policy)	16	L22
L21	l18 and ((quality adj4 service) or QoS or QOS)	4	L21
L20	L18 and l15	0	L20
L19	L18 and l16	0	L19
L18	command near8 (state adj4 value)	599	L18
L17	command near8 (stae adj4 value)	0	L17
L16	l15 and @AD<20000712	575	L16
L15	((quality adj4 service) or QoS or QOS) near8 (manage or management or managed)	1490	L15
L14	((quality adj4 service) or QoS or QOS) near8 (configurate or configuration) near8 command	7	L14
L13	(((command near8 (merge or merging or aggregate or aggregating))and ((command adj4 line adj4 interface) or CLI)) and @AD<20000712)	8	L13
L12	((command near8 (merge or merging or aggregate or aggregating)) and ((command adj4 line adj4 interface) or CLI))	14	L12
L11	(command near8 (merge or merging or aggregate or aggregating))	626	L11
L10	((abstract adj policy) and @AD<20000712)	16	L10
L9	((abstract adj4 policy) and @AD<20000712)	28	L9
L8	((abstract adj4 policy) near8 (command))	1	L8
L7	((CLI) and (abstract adj4 policy))	0	L7
L6	(((command adj4 line adj4 interface) or CLI) and (abstract adj4 policy))	1	L6
L5	(abstract adj4 policy)	43	L5
L4	(CLI)	2175	L4
L3	((command adj4 line adj4 interface) or CLI)	3360	L3
L2	((abstract adj4 policy) near8 (basic adj4 command))	0	L2
L1	((abstract adj4 policy) near8 (basic adj4 command))	0	L1

```
END OF SEARCH HISTORY
```

The Patent Office

PATENTS • DESIGNS • COPYRIGHT • TRADE MARKS

14

INVESTOR IN PEOPLE

Application N :	GB 0121847.8	**Examiner:**	Matthew Cope
Claims searched:	1-10	**Date of search:**	21 May 2002

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.T): G4A AUB, AUXX

Int Cl (Ed.7): H04L, G06F

Other: ONLINE: EPODOC, WPI, PAJ, INTERNET

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A,E	WO 01/44956 A1	RENSSELAER	-
A	WO 99/14891 A2	SCHMIDT	-
Y	US 5907696 A	STILWELL	2,5
X,Y	US 5726979 A	HENDERSON	1-10
Y	US 5594792 A	CHOURAKI - Column 2, line 19 onwards	1,2
X	www.ericsson.com/datacom/news/press_room/releases/19991012-0502.shtml		1 at least

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

WEST

(12) UK Patent Application (19) GB (11) 2 371 646 (13) A

(43) Date of A Publication 31.07.2002

(21) Application No 0121847.8

(22) Date of Filing 10.09.2001

(30) Priority Data
(31) 09659367 (32) 12.09.2000 (33) US

(71) Applicant(s)
Hewlett-Packard Company
(Incorporated in USA - Delaware)
3000 Hanover Street, Palo Alto, California 94304, United States of America

(72) Inventor(s)
Robert C Vacante
Peter Thomas Houck

(74) Agent and/or Address for Service
Carpmaels & Ransford
43 Bloomsbury Square, LONDON, WC1A 2RA, United Kingdom

(51) INT CL^7
G06F 17/50 , H04L 12/24

(52) UK CL (Edition T)
G4A AUB AUXX

(56) Documents Cited
WO 99/14891 A2
WO 01/44956 A1
US 5907696 A
US 5726979 A
US 5594792 A
www.ericsson.com/datacom/news/press_room/releases/19991012-0 502.shtml

(58) Field of Search
UK CL (Edition T) G4A AUB AUXX
INT CL^7 G06F , H04L
ONLINE: EPODOC, WPI, PAJ, INTERNET

(54) Abstract Title
Testing of policy prior to deployment in a policy-based network management system

(57) The invention provides a method of testing a policy prior to deployment in a policy-based network management system so that network traffic is less likely to be affected by errors in deployment. The method comprises creating an abstract policy, storing the abstract policy, assigning the abstract policy to a specific target device (140), transferring the assigned policy to an agent (130), translating the assigned policy into specific configuration commands by the agent, testing the configuration commands prior to deployment by the specific target device and deploying the configuration commands by the specific target device. The testing step may be performed on a backup device or a device emulator.

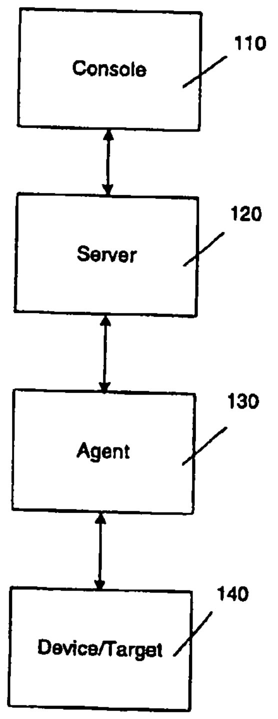

Fig. 1

GB 2 371 646 A

WEST

WEST

Generate Collection

L23: Entry 3 of 12 File: PGPB Jun 19, 2003

DOCUMENT-IDENTIFIER: US 20030115545 A1
TITLE: Dynamic display of data item evaluation

Abstract Paragraph (1):
A graphical user interface (GUI) is provided for use in conjunction with attribute evaluation utilizing computation rules and a combining policy. The graphical user interface displays a visual representation of the evaluation of attributes. The GUI displays a representation of the evaluation status of attributes and their associated computation rules during execution. The display may be updated dynamically to provide a user with a visual representation of the progression of attribute evaluation. The representation of evaluation status may include representations of the values assigned to, as well as the state of, data items and associated computation rules. States may be visually represented in the display in various ways, such as using various colors or labels associated with the data items and associated computation rules. Further, the particular data items and computation rules for which evaluation status is represented may be user selectable.

Summary of Invention Paragraph (5):
[0004] As an example of the processing of attributes in an object-focused system, consider a customer call center which receives customer telephone calls. An attribute that may be evaluated is call_priority, which will be assigned a value depending on various factors, such as previous calls from the customer, input already received from the customer identifying the reason for the call, the potential value of the call, etc. The factors are evaluated and the results of the evaluations are combined in some manner in order to determine the value to be assigned to the call_priority attribute. The manner in which the factors are evaluated, and the manner in which the results are combined, are defined by some business policy regarding the handling of calls. Once the business policy is articulated, it must be specified in a manner understandable to a computer system implementing the call center.

Summary of Invention Paragraph (6):
[0005] In accordance with the data evaluation technique described below, which is the subject of related copending U.S. patent application Ser. No. 09/253,274, entitled, Data Item Evaluation Based on the Combination of Multiple Factors, data items are evaluated in accordance with associated computation rules and a combining policy. The computation rules specify the factors to be taken into account in the evaluation of the data item, and how the factors are to be evaluated. The combining policy specifies how the results of the evaluation of the factors are to be combined in order to assign a value to the data item. The computation rules comprise a condition and a term. The evaluation of a computation rule consists of the evaluation of the condition, and if the condition is true, then contributing the term to a collection of values. The collection of values is then combined in accordance with the combining policy to produce a value which is assigned to the data item. The use of computation rules and a combining policy in accordance with this technique provides for a powerful and flexible technique for evaluating data items based on the combination of factors. Further, a program specification in accordance with this technique is easy to maintain and modify because the specification of how factors are to be evaluated are specified by the computation rules, and the specification of how the factor evaluation results are to be combined are specified by the combining policy. This separation results in a specification which is easier to understand, maintain, and modify.

Summary of Invention Paragraph (8):
[0006] The present invention provides an improved graphical user interface for use in conjunction with the data item evaluation technique utilizing computation rules and a combining policy. The graphical user interface displays a visual representation of the evaluation of data items

Brief Description of Drawings Paragraph (28):
[0036] FIG. 27 shows the typing rules for Combining Policy Language operators;

Detail Description Paragraph (12):
[0060] There are four types of modules in a DL specification. Declarative modules have a declarative semantics and contain sub-modules. Declarative modules are represented in the figures using rounded corner rectangles. Decision modules produce a single attribute value through a novel technique for aggregating and synthesizing information using computation rules and a combining policy which will be described in further detail below. Decision modules are represented in the figures using solid line hexagons. Flowchart modules are specified using a flowchart language to describe the processing. Flowchart modules are represented in the figures using rectangles with a cut corner. Foreign modules are implemented using some type of external function (e.g. database query or web server routine). Foreign modules are represented in the figures with a rectangle. Each of these types of modules will be described in further detail below.

Detail Description Paragraph (13):
[0061] As shown in FIG. 2, the Routing_To_Skill module 202 is made up of six sub-modules: Identify_Caller 210, Info_About_Customer 220, Info_From_Web 230, Promo_Selection 240, Routing_Decisions 250, and Calculate_Wrap_Up 260. Module 210 is represented as a rectangle with a cut corner indicating that it is a flowchart module. Modules 230 and 240 are represented as rectangles which indicates that these modules are foreign modules and obtain values for attributes by executing external functions. Modules 220 and 250 are represented as rounded corner rectangles, which indicates that these modules are declarative modules which are specified at a lower level using another DL specification. These types of modules are used to conveniently represent at a high level a substantial amount of lower level processing. Such high level representation makes it easier for a designer to visualize the overall function of a DL specification. Module 260 is represented as a hexagon, which indicates that this module is a decision module and evaluates a single attribute using computation rules and a combining policy as will be described in further detail below.

Detail Description Paragraph (25):
[0073] The DL specification further defining the Info_About_Customer module 220 (FIG. 2) is shown graphically in FIG. 5 and textually in FIG. 6. This Info_About_Customer module 220 is a declarative module and is therefore further defined in terms of sub-modules. The Get_Recent_Contacts_For_This_Customer module 504, the Get_Recent_Purchases_For_This_Customer module 508, the Get_Account_History_For_This_Customer module 512, and the Calculate_Cust_Value module 528 will always be evaluated because their respective enabling conditions 502, 506, 510, 526 are always true. It is noted that the graphical representation of these modules indicate that they are foreign modules. Each of these modules performs an external database retrieval function. If the attribute RECENT_CONTACTS has a state of VALUE, then the enabling condition 514 will be True and the Calculate_Frustration_Score module 516 will be evaluated. If the attribute RECENT_CONTACTS has state EXCEPTION, DISABLED, or FAILED, then the enabling condition 514 will be False and the Calculate_Frustration_Sc- ore module 516 will not be evaluated. If the attribute RECENT CONTACTS is in state UNINITIALIZED, then enabling condition 514 cannot yet be evaluated. Enabling conditions 518, 522 and 530 are evaluated in a similar manner. The modules 516, 520, 524, 528, and 532 are all represented as solid line hexagons, which indicates that these modules are decision modules and the processing of these modules is specified in terms of computation rules and a combining policy, as will be described in further detail below.

Detail Description Paragraph (28):
[0076] The DL specification further defining the Routing_Decisions module 250 (FIG. 2) is shown graphically in FIG. 9 and textually in FIG. 10. This Routing_Decisions module 250 is a declarative module and is therefore further defined in terms of sub-modules. The Ask_Reason_For_Call module 910 will be evaluated if the CUST_VALUE attribute has a value less than 7, as indicated by enabling condition 912. Module 910 is represented as a rectangle, which indicates that the module is a foreign module. This module 910 performs an IVR interaction asking the caller the reason for calling, and the reason is assigned to attribute IVR_CHOICE. Modules 920, 940, and 950 will all be evaluated because their associated enabling conditions 922, 942, and 952, are all True. Module 960 will not be evaluated if the attribute BUSINESS_VALUE is greater than 100 or if the attribute FRUSTRATION_SCORE is greater than 5. This enabling condition 962 illustrates that enabling conditions can also specify when a particular module is disabled, rather than specifying when it is enabled. Module 930 will be evaluated if the test in its

enabling condition 932 is True. The modules 920, 940, 950, and 960 are represented as hexagons, which indicates that these modules are decision modules and that their attributes are evaluated using computation rules and a combining policy. These modules will be described in further detail below. Module 930 is represented as a rectangle which indicates that this is a foreign module.

Detail Description Paragraph (30):
[0078] Referring back to FIG. 2, the final module in the Routing_To_Skill module 202 is the Calculate_Wrap_Up module 260. Calculate_Wrap_Up module 260 is graphically represented in FIG. 2 as a hexagon, which indicates that the module is a decision module and that the processing of the module is specified in terms of computation rules and a combining policy. The use of computation rules and a combining policy to evaluate attributes will now be described in detail.

Detail Description Paragraph (31):
[0079] In general, the format of computation rules and a combining policy within a DL specification is as follows:

Detail Description Paragraph (39):
[0087] Combining Policy:

Detail Description Paragraph (40):
[0088] Program in Combining Policy Language (CPL) or CPL function.

Detail Description Paragraph (43):
[0091] indicates that the term contributes to Attribute if the condition has a True value. Thus, each of the computation rules is evaluated to determine whether the condition is True or False. If the condition is True, then the term contributes to the attribute. If the condition is False, then the term does not contribute to the attribute. After each of the computation rules has been evaluated, the attribute is assigned a value based on the combining policy language (CPL) program, or the CPL function (where the CPL function is specified by a CPL program). Thus, the computation rules only contribute their terms to attributes, which is different from assigning a value to the attribute. The attribute is only assigned a value based on the combining policy (e.g. CPL program or CPL function). For example, a CPL function may indicate that the highest value of the contributed terms is to be assigned to the attribute or that the sum of all the contributed terms is to be assigned to the attribute.

Detail Description Paragraph (44):
[0092] It is noted that the use of computation rules and a combining policy has been described in the context of the object-focused workflow system. As such, computation rules and combining policies are used to assign values to attributes. It is to be understood, however, that the use of computation rules and combining policies is not limited to use in an object-focused system. More generally, computation rules and combining policies may be used to evaluate any type of data item, whether that data item is an attribute in an object-focused system, a variable in a procedural system, or some other type of data item.

Detail Description Paragraph (125):
[0173] The foregoing was a formal definition of the CPL language. Examples of how the language is used to implement combining policy will be given below in connection with the ongoing example.

Detail Description Paragraph (126):
[0174] Returning now to the description of the example application and FIG. 2, module 260 is specified in terms of computation rules and a combining policy, and is shown textually in FIG. 11. The computation rules are specified in line 25-39 of the DL specification. As defined in lines 21-22, the output attribute WRAP_UP is a set of tuples containing an attribute name and a value of the attribute. In effect, this attribute contains various attribute names and associated values for archival purposes. The first three rules (lines 26-31) are of the form "if true", so that they will always be True, and the values specified in those rules will always contribute to the attribute WRAP_UP. The computation rule on lines 32-35 will only contribute to the attribute WRAP_UP if the attribute WEB_DESTINATIONS is not empty. Similarly, the computation rule on lines 36-39 will only contribute to the attribute WRAP_UP if the value of CUST_REC.CARD_COLOR is "gold". The combining policy (wrap-up-cp) is specified in lines 40-41 and indicates that the contribution of all true rules are to be used. It is assumed that this combining policy is a predefined CPL function which is available for use by system designers. The CPL program defining this function is as follows:

Detail Description Paragraph (130):
[0178] Returning now to FIG. 5, the Calculate_Frustration_Score module 516 is represented as a hexagon, which indicates that the module is a decision module and that the processing of the module is specified in terms of computation rules and a combining policy. The DL specification for this module is shown in FIG. 15. The computation rules are shown in part in lines 7-19. There would likely be additional computation rules, but such rules are not shown because they are not necessary for an understanding of the relevant portions of FIG. 15. The input attribute to this module is RECENT_CONTACTS, which is a list as defined in FIG. 12, lines 4-10. Thus, the computation rule specified in lines 8-13 of FIG. 15 specifies that if the attribute RECENT_CONTACTS is defined for list element [1] (i.e., there is at least one recent contact specified in the RECENT_CONTACTS attribute), then the expression in lines 10-13 is evaluated and the result is contributed to the FRUSTRATION_SCORE attribute. Similarly, the computation rule specified in lines 14-19 of FIG. 15 specifies that if the attribute RECENT_CONTACTS is defined for list element [2] (i.e., there are at least two recent contacts specified in the RECENT_CONTACTS attribute), then the expression in lines 16-19 is evaluated and the result is contributed to the FRUSTRATION_SCORE attribute. Additional computation rules would likely be included in an actual implementation, but are not shown here. It is noted that in this example, any or all of the computation rules may evaluate to True, in which case the attribute FRUSTRATION_SCORE gets contributions from each of the true rules.

Detail Description Paragraph (131):
[0179] The combining policy for the Calculate_Frustration_Score module 516 is a CPL function, frustration-score-cp, as specified in lines 21-24 and indicates that the contributions of the true rules will be added together and rounded up to the next integer, up to a maximum of 10. The result is assigned to the attribute FRUSTRATION_SCORE. The CPL program defining this function is as follows:

Detail Description Paragraph (136):
[0184] The Calculate_Net_Profit_Score module 520 (FIG. 5) is represented as a hexagon, which indicates that this module is a decision module and that the processing of the module is specified using computation rules and a combining policy. The DL specification for this module is shown in FIG. 16. The computation rules are shown in lines 10-32. The input attributes to this module are RECENT_CONTACTS, RECENT_PURCHASES, ACCOUNT_HISTORY, and CUST_REC. The computation rules specified in lines 10-32 specify the contributions to the attribute NET_PROFIT_SCORE based on the input attributes. The combining policy specified in lines 33-35 is a CPL function, net-profit-score-cp, and indicates that the contributions of the true rules will be added together. The resulting sum is assigned to the attribute Net_Profit_Score. The CPL program defining this function is as follows:

Detail Description Paragraph (138):
[0186] The Calculate_Late_Payment_Score module 524 (FIG. 5) is specified using computation rules and a combining policy and the DL specification for this module is shown in FIG. 17. The computation rules specified in lines 7-19 specify the contributions to the LATE _PAYMENT_SCORE attribute based on the input attribute. The combining policy as specified in lines 20-22 is a CPL function, late-payment-score-cp, and indicates, that the contribution of a true rule will be assigned to the attribute LATE_PAYMENT_SCORE, or if there is no rule evaluating to a true value, then the LATE_PAYMENT_SCORE attribute will be assigned the default value of 0. It is noted that a constraint on this type of combining policy is that only one computation rule may evaluate to true. This constraint could be tested for during a pre-processing step to make sure that any possible evaluation will result in only one computation rule being true. The CPL program defining this function is as follows:

Detail Description Paragraph (141):
[0189] The Calculate_Cust_Value module 528 (FIG. 5) is specified using computation rules and a combining policy and the DL specification for this module is shown in FIG. 18. The computation rules specified in lines 9-19 specify the contributions to the CUST_VALUE attribute based on the input attributes. The combining policy as specified in lines 20-23 is a CPL function, calculate-cust-val-cp, and indicates that the contributions of rules with a true condition are added and rounded up, with a maximum of 100. The result is assigned to the attribute CUST_VALUE. If no rule evaluates to a true valve, then the CUST_VALUE attribute is assigned the default value 0. The CPL program defining this function is as follows:

Detail Description Paragraph (143):

[0191] The Calculate_Marketing_vs._Collections module 532 (FIG. 5) is specified using computation rules and a combining policy and the DL specification for this module is shown in FIG. 19. The computation rule specified in lines 8-16 specifies the contributions to the MARKETING_VS_COLLECTIONS attribute based on the input attributes. It is noted that in an actual implementation additional rules would likely be included. The combining policy as specified in lines 19-24 is a CPL function, marketing-vs-collection-cp, and indicates that the contribution of a true rule will be assigned to the attribute MARKETING_VS_COLLECTIONS- , or if there is no rule evaluating to a true value, then the MARKETING_VS_COLLECTIONS attribute will be assigned the default value "marketing". In this module, it is assumed that all computation rules (not all shown) have the effect of contributing the value "collect", and so there is no inconsistency if more than one rule has a true condition. The CPL program defining this function is as follows:

Detail Description Paragraph (146):
[0194] Module 920 is represented as a hexagon, which indicates that this module is a decision module and is specified using computation rules and a combining policy. The DL specification for module 920 is shown in FIG. 21. The computation rules specified in lines 12-30 specify the contributions to the BUSINESS_VALUE_OF_CALL attribute based on the input attributes. The combining policy as specified in lines 31-34 is a CPL function, business-value-of-call-cp, and indicates that the contribution of the rules with a true condition are added and rounded up with a default value of 0. The CPL program defining this function is as follows:

Detail Description Paragraph (149):
[0197] As shown in FIG. 9, all remaining sub-modules of the Routing_Decisions module 250 are represented as hexagons, which indicates that these modules are decision modules and are specified using computation rules and a combining policy. Turning now to the Calculate_Call_Priority module 940 module, the DL specification for this module is shown in FIG. 23. The computation rules specified in lines 8-20 specify the contributions to the CALL_PRIORITY attribute based on the input attributes. The combining policy as specified in lines 21-22 is a CPL function, call-priority-cp, and indicates that high value wins with a default value of 2. This means that the single highest contribution value for a true rule is the value that is assigned to the CALL_PRIORITY attribute. Thus, after all the computation rules are evaluated, there will be a collection of all the results, with the true rules contributing the value specified in the rule, and the false rules contributing .perp.. The combining policy will choose the highest value in the collection and assign that value to the output attribute CALL_PRIORITY. If no rule has a true condition, then the value of 2 is assigned to CALL_PRIORITY. The CPL program defining this function is as follows:

Detail Description Paragraph (152):
[0200] The DL specification for the Calculate_Skill module 950 is shown in FIG. 24. The computation rules specified in lines 11-40 specify the contributions to the SKILL attribute based on the input attributes. The combining policy as specified in lines 41-45 is a CPL function, skill_cp, and indicates a weighted sum policy with ties being broken by the given ordering. Referring back to the computation rules, each true rule will contribute both a value and a weight to the SKILL attribute. For example, if the computation rule on lines 14-15 is evaluated to True, then the value high_tier with a weight of 40 is contributed to the SKILL attribute. After all the computation rules are evaluated, the sum of each of the weights for a particular Value is computed, and the value that has the greatest weight is assigned to the SKILL attribute. If there is a tie between the weights of two different values, the value assigned to the SKILL attribute is determined by the ordering given by the combining policy. As an example, suppose the computation rule in line 28 and the computation rule in lines 32-35 are both evaluated to true. The computation rule in line 28 will contribute the value norm_tier_dom with a value of 30, and the computation rule in lines 32-35 will contribute the value norm_tier_dom with a value of 20. If these were the only two rules which contributed weights to the value norm_tier_dom, then the value norm-tier_dom would have a combined weight of 50 (30+20) when the combining policy was evaluated. If this combined weight of 50 for the value norm_tier_dom was the highest combined weight of any value, then the value norm_tier_dom would be assigned to the output attribute SKILL. The CPL program defining this function is as follows:

Detail Description Paragraph (153):
[0201] The DL specification for the Calculate_On_Queue_Promo module 960 is shown in FIG. 25. The computation rules specified in lines 8-13 specify the contributions to the ON_QUEUE_PROMO attribute based on the input attribute. The combining policy as specified in lines 14-15 is a CPL function, on-queue-promo-cp, and indicates first true

wins with a default of 0. In accordance with this combining policy, the contribution of the true rule will be assigned to the attribute ON_QUEUE_PROMO. As described above, a constraint on this type of combining policy is that one and only one computation rule may evaluate to true. The CPL program defining this function is as follows:

Detail Description Paragraph (167):
[0215] The above steps describe how a determination is made as to whether a module will be executed. With respect to how a module is executed, it depends on the type of module. If the module is other than a decision module and therefore specified in terms other than computation rules and a combining policy, then the module is executed as specified in the module (e.g. C++ program, database dip, flowchart, declarative module, etc.), and the details of such execution will not be described in detail herein. If the module is a decision module specified in terms of computation rules and a combining policy, then the module is executed as follows:

Detail Description Paragraph (173):
[0221] apply combining policy program to the collection of values to produce the attribute value

Detail Description Paragraph (349):
[0397] As described above, decision modules are evaluated using computation rules and a combining policy. In accordance with the present invention, a novel graphical user interface (GUI) is used to display a representation of the evaluation of decision modules. The GUI is particularly advantageous for understanding and debugging the semantics of the workflow system, and for understanding how different execution strategies affect the processing of different kinds of inputs.

Detail Description Paragraph (351):
[0399] The present invention may be implemented in connection with essentially any policy for evaluating decision attributes, i.e., those attributes that are evaluated as specified by a decision module. In order to illustrate the GUI most clearly, we do not use the policy for evaluating decision attributes described above. Instead, the GUI is described using an execution policy that is eager with respect to the evaluation of computation rule conditions and computation rule terms. The contribution rule terms are also referred to as "contributions" because, as described above, these terms contribute values to an attribute if the condition is true. Given the description herein, one skilled in the art could readily implement the invention in connection with other execution policies.

Detail Description Paragraph (352):
[0400] In order to describe the eager execution policy for decision attributes, we modify the notion of snapshot used earlier, in two ways. The first modification is to restrict the set of states that decision modules can be in, and the second modification is to permit computation rule conditions and contributions to be evaluated in an eager fashion.

Detail Description Paragraph (357):
[0405] 2. Dec is a set of pairs {(Rules.sub.A, CP.sub.A).vertline.A is a decision attribute}, where for each decision attribute A, Rules.sub.A is the set of computation rules in the module outputting A, and CP.sub.A is the combining policy for the module outputting A.

Detail Description Paragraph (375):
[0423] For the purposes of describing the GUI, it is not necessary to use a specific algorithm or policy for executing a declarative workflow. We assume here that execution of workflow S begins with a snapshot such that all source attributes are stable, all internal modules are in state UNINITIALIZED (or READY, for decision modules), and all computation rules are in state READY. Then a sequence of snapshots is constructed, each one a strict extension of the previous one. Execution halts when a terminal snapshot is reached.

Detail Description Paragraph (378):
[0426] Referring to FIG. 38, the display is in a grid format, with the rows labeled with numbers and the columns labeled with letters. The intersection of a row and a column defines a cell. Each column of the display corresponds to an attribute of the INFO_ABOUT_CUSTOMER module. The first two rows provide information about how the attributes are computed. Row 1 indicates the name of the module computing the attribute. For ease of cross-reference, row 1 of FIG. 38 includes the corresponding call-out identification of the module from FIG. 5. Such call-out numbers would not be

included in an actual embodiment of the invention. Row 2 indicates the manner of computation. For non-decision modules, row 2 indicates the type of the module (e.g., "foreign"). For decision modules, row 2 displays a short description of the combining policy of the module. This short description could be specified, for example, in the textual description of the combining policy. As an alternative, row 2 could display the name of the function specifying the combining policy for the module. The attribute names are shown in row 3.

Detail Description Paragraph (387):
[0435] We now describe one way that the processing for supporting the GUI could be incorporated into the basic algorithm of FIG. 34. Because this algorithm views the execution of decision modules as external "black boxes", the illustration here does not include a display of the incremental evaluation of computation rules. The Initialization step of FIG. 40A could be included at the end of part 3406 of FIG. 34A. The Iteration phase of FIGS. 40A and 40B could be included in section 3414 of FIG. 34B, just after section 3420. In this case, the Iteration phase would be applied multiple times, once for each relevant event that occurs during execution of section 3422. Alternatively, the Iteration phase of FIGS. 40A and 40B could be included (a) into section 3414 of FIG. 34B just after section 3418 and (b) into section 3422 of FIGS. 34B and 34C just after each occurrence of a command that assigns a state value to an attribute (i.e., a value for .sigma.[C] for some attribute C). Based on this description it would be clear to one skilled in the art how to incorporate processing to support the GUI into the extended algorithm of FIG. 35, and into algorithms that support execution of DL specifications that are eager with respect to the evaluation of computation rule conditions and/or computation rule terms.

Detail Description Paragraph (395):
[0443] 4. User control over visual layout: Permit the user to hide or expose selected columns or rows of the display. Also, permit the user to click on cells to display more information about them. For example, clicking on a rule cell could result in the display of a pop-up window showing the rule. Clicking on a cell in row 2 could result in the display of the CPL program specifying the combining policy associated with that cell.

CLAIMS:

1. In a computer system which implements an object-focused workflow system and which evaluates at least one attribute by evaluating a plurality of computation rules corresponding to the at least one attribute to produce a collection of values and assigning a final value to said attribute by applying at least one combining policy to said collection of values, a method for displaying a representation of the evaluation of said at least one attribute on a display screen comprising the steps of: displaying a representation of the evaluation status of said at least one attribute, wherein said at least one attribute corresponds to an object in said object-focused workflow system; displaying a representation of the evaluation status for each of said plurality of computation rules, each of said computation rules defining whether a value should be contributed to said at least one attribute; and displaying a representation of said at least one combining policy, each of said combining policies defining a manner in which said contributed data values are combined in order to determine a final value for said at least one attribute.

5. The method of claim 1 wherein said representation of the evaluation status of said at least one attribute includes said final value assigned to said at least one attribute, wherein said final value is displayed after said combining policy is applied to said collection of values.

7. The method of claim 1 further comprising the step of: displaying a representation of said combining policy.

11. The method of claim 1 further comprising the step of: displaying representations of said computation rules in a structure which represents said combining policy.

12. The method of claim 1 wherein a representation of a selected combining policy comprises a textual description of said selected combining policy.

13. The method of claim 1 wherein a representation of a selected combining policy comprises a name of a function specifying said selected combining policy.

14. The method of claim 1, wherein: said at least one attribute comprises a plurality

of attributes; the object-focused workflow system comprises an instance of a workflow specification, whereby the computer system implements said instance of said workflow specification; the step of displaying a representation of the evaluation status of said at least one attribute further comprises the step of displaying a representation of an evaluation status for each of the plurality of attributes; the method further comprises the steps of: initializing said representations of evaluation statuses of the plurality of attributes; initializing said representations of said evaluation statuses of said computation rules associated with each of said plurality of attributes; executing said instance of said workflow specification; receiving at least one event occurring during execution of said instance of said workflow specification; and based on said at least one event, performing one or more of the following steps: (i) updating said representation of the evaluation status of said plurality of attributes; (ii) updating said representations of the evaluation statuses of said computation rules associated with each of said plurality of attributes; and (iii) updating said representation of said at least one combining policy.

20. In a computer system which implements an object-focused workflow system and which evaluates at least one attribute by evaluating a plurality of computation rules corresponding to the at least one attribute to produce a collection of values and assigning a final value to said attribute by applying at least one combining policy to said collection of values, an improvement for displaying a representation of the evaluation of said at least one attribute on a display screen comprising: a memory storing computer program instructions executable by a processor of said computer system, said computer program instructions defining the steps of: displaying a representation of the evaluation status of said at least one attribute, wherein said at least one attribute corresponds to an object in said object-focused workflow system; displaying a representation of the evaluation status for each of said plurality of computation rules, each of said computation rules defining whether a value should be contributed to said attribute; and displaying a representation of said at least one combining policy, each of said combining policies defining a manner in which said contributed data values are combined in order to determine a final value for said attribute.

WEST

Generate Collection

L6: Entry 1 of 1 File: USPT Nov 19, 2002

DOCUMENT-IDENTIFIER: US 6484261 B1
**** See image for Certificate of Correction ****
TITLE: Graphical network security policy management

Abstract Text (1):
A method of establishing a representation of an abstract network security policy is disclosed. The representation is established in the form of a decision tree that is constructed by assembling graphical symbols representing policy actions and policy conditions. A user modifies properties of the graphical symbols to create a logical representation of the policy. Concurrently, the logical representation is transformed into a textual script that represents the policy, and the script is displayed as the user works with the logical representation. When the policy representation is saved, the script is translated into machine instructions that govern the operation of a network gateway or firewall. The policy representation is named. The policy representation may be applied to other network devices or objects by moving an icon identifying the representation over an icon representing the network device. Policies, network objects, and network services are stored in the form of trees.

Brief Summary Text (4):
Administrators of computer networks generally think of network security in terms of abstract security policies. The administrators design the security policies to protect their organization's information resources against threats that may compromise the confidentiality, integrity, or availability of sensitive data. However, the way that people conceptualize security policies does not match the way that they must implement them using conventional, rule-based security policy models.

Brief Summary Text (8):
Currently, security policies are generally prepared using an ordered list of rules. In past approaches, the network devices are designed to interact with operating systems having text-based, command-line interfaces. Because of these interfaces, administrators had to learn the command sets that controlled how the devices operated. The command sets were, and still are, cryptic and difficult to use. The command sets differ from one network device vendor to the next. Moreover, the relationship between different lines of a command set may cause problems; a previous rule may affect the execution of all later rules, or even prevent their use. These inter-relationships are difficult to remember or track.

Brief Summary Text (17):
In addition, there are several problems associated with managing and maintaining the representations of security policies generated by use of the icon interface. The representations are difficult to conceptualize and relate to an abstract security policy. It is difficult to verify that security policies are applied correctly and consistently to all network objects. It is difficult to define exceptions and changes to security policies. The past approaches do not generally distinguish between users and network objects, and do not permit security policies to be ported to other locations.

Brief Summary Text (24):
Thus, there is a need for a method or mechanism to construct a network security policy without the use of a list of rules. In particular, there is a need for a way to construct a network security policy that is easily understood by a network administrator, that avoids the use of router-based rule sets, and in which an abstract security policy is easily correlated with a representation of the policy.

Detailed Description Text (73):
The administration component 206 provides mechanisms for constructing representations of abstract network security policies. After a security policy is constructed, it is

represented in the policy tree 316 as a named policy. The security policy is applied to a node of the network tree 314 by dragging the icon representing the policy from the policy tree 316 to the node in the network tree 314. When the network tree 314 is saved, the administration component 206 constructs instructions to the firewall that cause the firewall to enforce the security policies that have been defined.

Detailed Description Text (156):
The mechanisms described above are used to define a graphic display, a script, and firewall instructions that reflect abstract network security policies. For example, consider a hypothetical company, Acme Corporation, which has employees organized as engineering, marketing, and administration departments. The administration employees do not need access to the Internet except to use electronic mail (email). The engineering group uses anonymous file transfer protocol (FTP) to transfer files to satellite offices, accesses USENET newsgroups for solutions, and browses World Wide Web sites to look for software patches. The marketing group uses the Web for market research. Acme's standard security policy prohibits Acme systems from receiving Active-X or Java applets that can contain malicious code.

WEST

☐ Generate Collection

L10: Entry 1 of 16 File: USPT Oct 15, 2002

DOCUMENT-IDENTIFIER: US 6466984 B1
TITLE: Method and apparatus for policy-based management of quality of service treatments of network data traffic flows by integrating policies with application programs

Abstract Text (1):
A method and apparatus for policy-based management of quality of service treatments of network data traffic flows by integrating policies with application programs are described. In one embodiment, a quality of service value is selectively associated with a flow of information generated by an application program and directed to a network device. Mappings representing an abstract policy and associating a pre-determined network quality of service with a traffic flow type of the flow of information and with an application program are created and stored in a repository that is accessible by the application program. The mappings are converted into one or more settings of the network device. The policy is enforced at the network device in response to receiving traffic from the application program that matches the traffic flow type. The settings may be Differentiated Services Code Points or may be RSVP+ messages. Policies may be represented by statements stored in a directory schema. Each policy statement is represented by nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising one of the quality of service treatments. The nodes start at a root node having a distinguished name in the directory.

Application Filing Date (1):
19990702

Brief Summary Text (32):
The foregoing objects and advantages, and other objects and advantages that will become apparent from the following description, are achieved by the present invention, which comprises, in one embodiment, a method of selectively associating a quality of service value with a flow of information generated by an application program and directed to a network device. The method involves creating one or more mappings, each mapping representing an abstract policy and associating a pre-determined network quality of service with a traffic flow type of the flow of information and with an application program. The mappings are stored in a repository that is accessible by the application program. The mappings are converted into one or more settings of the network device, which enforces the policy in response to receiving traffic from the application program that matches the traffic flow type.

CLAIMS:

1. A method of selectively associating a quality of service with a flow of information generated by an application program and directed to a network device, comprising the steps of: creating one or more mappings, each mapping representing an abstract policy and associating a pre-determined network quality of service with a traffic flow type of the flow of information and with an application program; storing the mappings in a repository that is accessible by the application program; converting the mappings into one or more settings of the network device that may be used by the network device to enforce the policy at the network device in response to receiving traffic from the application program that matches the traffic flow type.

9. A method as recited in claim 1, wherein the abstract policy for each mapping is determined by creating and storing one or more policy statements in a repository, wherein each policy statement associates a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

10. A method as recited in claim 1, wherein the abstract policy for each mapping is

determined by creating and storing one or more policy statements in a repository, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

11. A method as recited in claim 1, wherein the abstract policy for each mapping is determined by creating and storing one or more policy statements in a directory, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment, and wherein the plurality of nodes is coupled to a root node having a distinguished name in the directory.

13. A method as recited in claim 1, wherein the abstract policies are enforced by creating and storing messages requesting an operating system function to modify a packet of the traffic flows using a policy element that requests a different operating system function according to the operating system then in use; and at the network device, in response to receiving traffic from the application program that matches the traffic flow type and in response to the operating system function, modifying the packet to activate a quality of service treatment of the network device.

15. A computer-readable medium carrying one or more sequences of instructions for selectively associating a quality of service with a flow of information generated by an application program and directed to a network device, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of: creating one or more mappings, each mapping representing an abstract policy and associating a pre-determined network quality of service with a traffic flow type of the flow of information and with an application program; storing the mappings in a repository that is accessible by the application program; converting the mappings into one or more settings of the network device that may be used by the network device to enforce the policy at the network device in response to receiving traffic from the application program that matches the traffic flow type.

23. A computer-readable medium as recited in claim 15, wherein the abstract policy for each mapping is determined by creating and storing one or more policy statements in a repository, wherein each policy statement associates a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

24. A computer-readable medium as recited in claim 15, wherein the abstract policy for each mapping is determined by creating and storing one or more policy statements in a repository, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

25. A computer-readable medium as recited in claim 15, wherein the abstract policy for each mapping is determined by creating and storing one or more policy statements in a directory, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment, and wherein the plurality of nodes is coupled to a root node having a distinguished name in the directory.

27. A computer-readable medium as recited in claim 15, wherein enforcing one of the abstract policies comprises: requesting an operating system function to modify a packet of the traffic flows using a policy element that requests a different operating system function according to the operating system then in use; at the network device, in response to receiving traffic from the application program that matches the traffic flow type and in response to the operating system function, modifying the packet to activate a quality of service treatment of the network device.

28. A system for selectively associating a quality of service with a flow of information generated by an application program and directed to a network device, comprising: a policy manager that creates one or more mappings, each mapping representing an abstract policy and associating a pre-determined network quality of service with a traffic flow type of the flow of information and with an application program, wherein the mappings are stored in a repository that is accessible by the application program; a local storage element that converts the mappings into one or more settings of the network device that cause the network device to enforce the policy in response to receiving traffic from the application program that matches the traffic

flow type.

39. An apparatus for selectively associating a quality of service with a flow of information generated by an application program and directed to a network device, comprising: means for creating one or more mappings, each mapping representing an abstract policy and associating a pre-determined network quality of service with a traffic flow type of the flow of information and with an application program; means for storing the mappings in a repository that is accessible by the application program; means for converting the mappings into one or more settings of the network device that may be used by the network device to enforce the policy at the network device in response to receiving traffic from the application program that matches the traffic flow type.

47. An apparatus as recited in claim 39, wherein the abstract policy for each mapping is determined by means for creating and storing one or more policy statements in a repository, wherein each policy statement associates a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

48. An apparatus as recited in claim 39, wherein the abstract policy for each mapping is determined by means for creating and storing one or more policy statements in a repository, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

49. An apparatus as recited in claim 39, wherein the abstract policy for each mapping is determined by means for creating and storing one or more policy statements in a directory, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment, and wherein the plurality of nodes is coupled to a root node having a distinguished name in the directory.

51. An apparatus as recited in claim 39, wherein the abstract policies are enforced by: means for creating and storing messages requesting an operating system function to modify a packet of the traffic flows using a policy element that requests a different operating system function according to the operating system then in use; and at the network device, in response to receiving traffic from the application program that matches the traffic flow type and in response to the operating system function, means for modifying the packet to activate a quality of service treatment of the network device.

52. An apparatus for selectively associating a quality of service with a flow of information generated by an application program and directed to a network device, comprising: a network interface; a processor coupled to the network interface and receiving information from the network interface; a computer-readable medium accessible by the processor and comprising one or more sequences of instructions which, when executed by the processor, cause the processor to carry out the steps of: creating one or more mappings, each mapping representing an abstract policy and associating a pre-determined network quality of service with a traffic flow type of the flow of information and with an application program; storing the mappings in a repository that is accessible by the application program; converting the mappings into one or more settings of the network device that may be used by the network device to enforce the policy at the network device in response to receiving traffic from the application program that matches the traffic flow type.

60. An apparatus as recited in claim 52, further comprising instructions for determining the abstract policy for each mapping by performing the step of creating and storing one or more policy statements in a repository, wherein each policy statement associates a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

61. An apparatus as recited in claim 52, further comprising instructions for determining the abstract policy for each mapping by performing the step of creating and storing one or more policy statements in a repository, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment.

62. An apparatus as recited in claim 52, further comprising instructions for

determining the abstract policy for each mapping by performing the step of creating and storing one or more policy statements in a directory, wherein each policy statement is represented by a plurality of nodes that represent a condition of one of the traffic flows, an operator, an operand, and an action comprising a quality of service treatment, and wherein the plurality of nodes is coupled to a root node having a distinguished name in the directory.

64. An apparatus as recited in claim 52, further comprising instructions for enforcing the abstract policies by performing the steps of: creating and storing messages requesting an operating system function to modify a packet of the traffic flows using a policy element that requests a different operating system function according to the operating system then in use; and at the network device, in response to receiving traffic from the application program that matches the traffic flow type and in response to the operating system function, modifying the packet to activate a quality of service treatment of the network device.

WEST

☐ Generate Collection

L10: Entry 3 of 16 File: USPT May 21, 2002

DOCUMENT-IDENTIFIER: US 6393473 B1
TITLE: Representing and verifying network management policies using collective constraints

Application Filing Date (1):
19981218

Brief Summary Text (5):
The behavior of some network management applications or equipment may be governed by one or more abstract policies. A network management policy expresses a business goal for use of the network; the network management application can convert the policy into instructions to network devices, such as switches, routers, and other hardware and software, to implement the policy. An example of a policy is: "All administrative assistants may use the World Wide Web only between 11 a.m. and 3 p.m., Monday through Friday." A system that can receive and act on such policies is sometimes called a policy-based network management system.

Detailed Description Text (66):
The constraints between attributes or variables are called "instance constraints." Collective constraints also are called "group constraints." A group constraint may reflect an abstract policy, such as "only one type of traffic is allowed on an interface." Preferably, instance constraints and group constraints are stored in one or more database tables that are organized by variable and value.

Detailed Description Text (76):
These variables and values are stored in the initialization phase in block 202. Assume further that the system, or one or more abstract policies that are used to manage network devices of a managed network, defines the following constraints:

Detailed Description Text (242):
The new constraint (Hub, Deny) does not contradict (Hub, Permit). The network management system interprets these two constraints as associations and not as required pairings. Thus, (Hub, Permit) expresses the abstract policy that "Hubs are compatible with Permit Internet Access." Similarly, (Hub, Deny) means "Hubs are compatible with Deny Internet Access." In contrast, the rule IF Hub THEN Deny contradicts the rule IF Hub THEN Permit.

WEST

Generate Collection

L10: Entry 14 of 16 File: EPAB Dec 20, 2000

DOCUMENT-IDENTIFIER: EP 1061431 A2
TITLE: Configuring computer systems

Abstract Text (1):
CHG DATE=20010202 STATUS=O>&ORDF;&ORDF;&ORDF;&ORDF;An apparatus (22,44) is described for use in generating configuration information for a computer system (12) employing hierarchical entities. A policy template (24) is employed which contains a definition of an abstract high-level policy, for the configuration of the system, and permitted refinements to that policy, the definition referring to a plurality of the entities. An information and system model (16) contains information about the computer system and its environment including the entities referred to in the high-level policy definition, the hierarchy thereof and non-hierarchical relations between the entities. A policy authoring engine (26) refines the high-level policy definition with reference to the permitted refinements thereto and the stored information about the entities to which the high-level policy definition relates in order to produce a refined policy definition. In doing this, the engine presents refinement options to a user (10) via a user interface (28) and refines the high-level policy definition in dependence upon options selected by the user via the user interface. Some of the entities stored in the model (16) may be abstract entities, but with pointers to data in the computer system representing an instance of that abstract entity. The refined policy may be in terms of a policy context, referring to unbound entities, and a policy statement. A policy deployer (20) stores rules for interpreting the policy statement as instructions executable by the computer system and is operable, with reference to the information and system model (16), to bind the unbound entities in the policy context to instances of those entities, and, with reference to the stored rules, to interpret the policy statement into a series of instructions to the computer system referring to the bound instances or derivatives of them. The apparatus facilitates the refinement of abstract policies and implementation of the refined policies. ract policies and

Application Date (1):
20000505